Handle With Care! A Mechanistic Case Study of DPO Out-of-Distribution Extrapolation

Stanford CS224N Custom Project

Ryan Park
Department of Computer Science
Stanford University
rypark@stanford.edu

Abstract

Though direct preference optimization (DPO) is popular for model alignment, it tends to produce out-of-distribution (OOD) behavior in language models. Because this problem has only been recently noted, it is not well-studied or understood. By empirical analysis in a synthetic setting, we seek to characterize DPO's OOD behavior, exploring if it exhibits signs of sudden generalization ("grokking") that others have observed with a supervised fine-tuning. We find that once DPO strays out-of-distribution, it never comes back. Furthermore, we relate the OOD-ness of DPO policies to KL divergence from the reference model, and give a mechanistic explanation for how this OOD behavior arises. These results offer practical insights for aligning models with DPO. Given DPO's recent popularity in LLM alignment, these insights are valuable to practitioners seeking to build NLP systems.

Mentor: Archit Sharma (*solo project, no external collaborators*)

1 Introduction

Recently, large language models (LLMs) have been successful in understanding, conversing with, and obeying human instruction. This success is in large part due to good alignment methods, which are algorithms that take a pretrained language model and "align" them with intended use cases downstream. Many standard alignment methods, such as RLHF (Ouyang et al., 2022), use reinforcement learning (RL) to ensure LLMs effectively follow user instructions and obey guidelines. These methods require practitioners to explicitly model a reward function, which indicates the quality of LLM generations. However, in recent work, an RL-free alignment method called direct preference optimization (DPO) has been shown to be a simple and effective counterpart to RLHF-like methods (Rafailov et al., 2023). By deriving a transformation between LLM policy and optimal reward function, DPO avoids the need for RL algorithms and is thus much easier to use. Many of the leading open-source models are fine-tuned with DPO[1].

Though successful, DPO exhibits strange training dynamics. One fundamental issue with DPO is its tendency to quickly shift probability mass towards out-of-distribution (OOD) trajectories (Rafailov et al., 2024). This OOD behavior can be understood as an extreme type of overfitting, where DPO policies fit behaviors not even present in the train set. This problem is quite recent, and as such, there is not much work attempting to characterize or understand why/when/how this happens. Answering these questions would 1) help inform more principled upgrades to standard DPO, and 2) provide practical knowledge for practitioners attempting to fine-tune models with DPO.

Here, we empirically study this the issue of OOD extrapolation through a case study with a synthetic dataset. We focus on DPO with the standard Transformer (Vaswani et al., 2023), since this is most applicable to modern NLP. Our findings are as follows:

[1]https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

OOD extrapolation is irreversible. While most practitioners only run DPO for a couple epochs, it is worth considering whether DPO can exhibit "grokking" - i.e., sudden abilities to generalize after a large period of overfitting (Nanda et al., 2023), (Liu et al., 2022). We find that this is not the case.

Good DPO policies fall in similar KL regions. By looking only at KL divergence from initial policy (and letting the β parameter/number of steps vary freely), we can estimate which DPO policies will stay in-distribution and which will veer OOD.

Attention weights drive OOD extrapolation. We argue that DPO tends to focus its updates on attention weights in a subtle manner, and that we can distinguish between "good" and "bad" DPO policies by isolating the impact of attention weights on policy KL divergence.

Along the way, we develop new evaluation tools for understanding DPO OOD behavior. We hope these results provide practical insights on DPO's strange training behavior, and are useful for those trying to use DPO for model alignment.

2 Related Work

DPO overview. Given a reference policy $\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})$ and preference dataset $\mathcal{D}$, DPO starts with the standard KL-regularized RL objective (Ouyang et al., 2022):

$$\max_{\pi_\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\mathbf{y}\sim\pi_\theta(\mathbf{y}|\mathbf{x})}\left[r_\phi(\mathbf{x},\mathbf{y})\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi_\theta(\mathbf{y}\mid\mathbf{x}) \mid\mid \pi_{\text{ref}}(\mathbf{y}|\mathbf{x})\right] \tag{1}$$

where β controls the KL divergence from π_{ref}, π_θ is the DPO-trained policy, and r_θ is a reward function over $\mathcal{D}$. By deriving a transformation between optimal reward and policy, DPO attempts to optimize this objective via the following loss (Rafailov et al., 2023):

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta;\pi_{\text{ref}}\right) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta\left(y_w\mid x\right)}{\pi_{\text{ref}}\left(y_w\mid x\right)} - \beta\log\frac{\pi_\theta\left(y_l\mid x\right)}{\pi_{\text{ref}}\left(y_l\mid x\right)}\right)\right] \tag{2}$$

where σ is the sigmoid function, y_w is the preferred completion, and y_l is the dispreferred completion. Importantly, from Eq. 1, DPO derives a reward reparametrization of the form:

$$r_\theta(\mathbf{y},\mathbf{x}) = \beta\log\frac{\pi_\theta(\mathbf{y}\mid\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}\mid\mathbf{x})} \tag{3}$$

which we call the DPO "implicit reward" here. This is useful for analyzing the behavior of DPO.

DPO failures. DPO is particularly unstable after even moderate amounts of training. Prior research shows that DPO policies' downstream performance collapses after only a few hundred update steps (Rafailov et al., 2023), (Guo et al., 2024), (Rafailov et al., 2024). Some hypothesize that this collapse is due overfitting along spurious features like length (Park et al., 2024). Others argue that offline methods like DPO fail when the reward peak is far from the reference model, which may be the case for some alignment tasks (Tajwar et al., 2024). While some solutions have been proposed, such as using a reference-free version of DPO (Meng et al., 2024) and extending DPO to the online setting (Guo et al., 2024), the issues driving DPO OOD behavior are not fully understand.

Mechanistic interpretability and grokking. Nanda et al. (2023) and Liu et al. (2022) present examples of principled interpretability studies that inspired this work. Nanda et al. (2023) shows that after a long period of training, a 1-layer Transformer suddenly switches from a memorizing algorithm to a generalized algorithm in several toy settings (i.e., it "groks"). Liu et al. (2022) argues that grokking behavior is due to effective representation learning in the embedding space. These observations motivate this work, which partially explores whether DPO can grok as well.

3 Approach

3.1 Overview

To understand the dynamics characterizing OOD extrapolation, we design a synthetic toy dataset modeling the identity function via preference pairs on the number line. We run the standard DPO pipeline on this dataset with a 1-layer decoder-only Transformer (Vaswani et al., 2023). Let π represent this model. To compute π_{ref}, we run supervised fine-tuning (SFT) according to Eq. 4.

$$\mathcal{L}_{\text{SFT}}\left(\pi_{\text{ref}}\right) = -\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log \pi_{\text{ref}}\left(y \mid x\right)\right] \tag{4}$$

After SFT, we run DPO according to Eq. 2. We do not pretrain π before running SFT. The goal of this approach is to provide a fast and interpretable setting in which we can query DPO behavior.

We run three experiments. First, to see if DPO exhibits grokking, we run DPO for an abnormally large amount of epochs (100), analyzing OOD behavior. We recognize this is non-standard, but note that grokking (if possible) usually requires an exceedingly long training horizon.

Next, we run a sweep over 30 β values, running DPO for 3 epochs and sampling from a test set 10 times per epoch, producing ~3.5M individual samples. We consider how looking at KL divergence via Eq. 6 admits a better understanding of when DPO goes OOD and when it doesn't.

Finally, we run mechanistic experiments to analyze what parts of the trained policies (Transformers) are responsible for OOD behavior, and how DPO modifies them during training.

3.2 Synthetic dataset

This dataset is inspired by Nanda et al. (2023), Liu et al. (2022), and Im and Li (2024), but is original. It attempts to model an identity function on a subset of the number line. Let S be a set of positive integers, and W be a fixed window size. For all $s \in S$, for adjacent numbers $y_1, y_2 \in [s-W, s+W]$, let $y_1 \succ y_2$ if $|y_1 - s| < |y_2 - s|$ and vice versa ($y_1 \succ y_2$ denotes y_1 is preferred over y_1). The dataset $\mathcal{D}$ is the set of all such (s, y_1, y_2) tuples (for SFT, we use (s, y_1)). Intuitively, this dataset is a set of preferences clustered around each $s \in S$, incentivizing the model to emulate the identity function by preferring numbers closest to s. We split $\mathcal{D}$ into a train and test set by randomly partitioning on s, so tuples with the same prompt stay together. We consider s to be the prompt and y_n to be the response.

For the grokking experiment, we use $W = 5$ and $S = [1, 99]$ (337 train, 145 test examples). For other experiments, we use $W = 10$ and $S = [100, 500]$ (3754 train, 3755 test examples).

3.3 Experimental details

We use a 1-layer decoder-only Transformer, with $d_{\text{model}} = 128$ and 4 attention heads (103K parameters). We use learned positional embeddings, LayerNorm and residual connections throughout, as well as the standard post-attention 2-layer GeLU feed-forward network (see Vaswani et al. (2023) for full architecture details, excluded here for brevity). Its input is a tokenized representation of $s \in S$, and its output is a sequence of digits. Our implementation is from scratch in PyTorch.

We tokenize each dataset tuple (s, y) via the format "`ABC=XYZ.`" where `=` and `.` are special tokens, and each digit is its own token (so the vocabulary size is 12, including a pad token). We do not pad s, instead right-padding the concatenated sequence $s|y$ batchwise. Sampling is done greedily.

For the grokking experiments, SFT and DPO training is done for 100 epochs with AdamW (learning rate 10^{-4}, standard weight decay), batch size 512, and gradient clipping threshold of 5. AdamW is used since explicit regularization helps SFT models grok (Nanda et al., 2023). DPO models were trained with $\beta \in \{0.01, 0.02, 0.1, 0.2, 0.5\}$.

For other experiments, SFT was run for 10 epochs with Adam (learning rate 10^{-3}), and DPO run for 3 epochs with RMSprop (learning rate of 5×10^{-5}), all with batch size 16 and no gradient clipping. 30 β values were chosen to be spaced evenly on a log scale in $[10^{-2}, 10^{0}]$. 30 model samples on the full test set were taken uniformly during 3 DPO epochs. All training was done on a Macbook Air M1.

3.4 Evaluation metrics

The main tool of analysis in this study is the KL divergence, which for two discrete probability distributions $P(x)$ and $Q(x)$ is defined as

$$\mathbb{D}_{\text{KL}}\left[P \mid\mid Q\right] = \sum_{x\sim P} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{5}$$

We can relate the DPO implicit reward given in Eq. 1 to this quantity by noting that

$$\mathbb{D}_{\text{KL}}\left[\pi_\theta \mid\mid \pi_{\text{ref}}\right] = \mathbb{E}_{\mathbf{y}\sim\pi_\theta|\mathbf{x}}\left[\log\frac{\pi_\theta(\mathbf{y}\mid\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}\mid\mathbf{x})}\right] = \mathbb{E}_{\mathbf{y}\sim\pi_\theta|\mathbf{x}}\left[\frac{1}{\beta}r_\theta(\mathbf{y},\mathbf{x})\right] \tag{6}$$

This is fairly standard in the DPO literature (Rafailov et al., 2023). Given a model M, we define KL(M) as the scaled reward in Eq. 6 evaluated over the test set prompts.

Again using the DPO implicit reward, we define the DPO classification accuracy Acc(y_w, y_l, x) as true if $r_\theta(y_w, x) > r_\theta(y_l, x)$ and false otherwise. Similarly, let Margin$(y_w, y_l, x) = r_\theta(y_w, x) - r_\theta(y_l, x)$. These are from Rafailov et al. (2023) as well.

Next, we consider the sequence regression error, as well as our measure for OOD-ness. Let Y be a string of tokens sampled from the Transformer given x. We define Error(Y, x) as $|Y - x|$ if Y is a valid integer, or $\max S - \min S$ if Y cannot be parsed. Additionally, we define the binary measure OOD(Y, x) as true if Y is not parseable or $Y \notin [x - W, x + W]$, and false otherwise. We define OOD-ness in this way since in the train set, every prompt-response tuple (x, y) obeys $y \in [x - W, x + W]$. So any y not in this range is not in-distribution.

For the mechanistic experiments, we define the weight delta between two models, which is useful for measuring how much a model has changed during training. Given models M_1, M_2 with the same set of parameters Params, and optionally $Q \subseteq$ Params (some subset of all the parameters), let

$$\text{WeightDelta}(M_1, M_2) = \sum_{P\in\text{Params}} \|M_1[P] - M_2[P]\|_F \tag{7}$$

$$\text{FracWeightDelta}(M_1, M_2, Q) = \frac{1}{\text{WeightDelta}(M_1, M_2)} \sum_{P\in Q} \|M_1[P] - M_2[P]\|_F \tag{8}$$

Note that FracWeightDelta is useful for measuring how much a part of a model (e.g., attention weights) has changed relative to the total model change after DPO. Note that the WeightDelta metric is a simple and standard way to measure model difference.

Finally, we consider model surgery as a useful way to measure the contribution of a particular subset of weights. Given two models M_1, M_2, consider replacing some subset of the parameters (Q) in M_1 with the equivalent weights from M_2, and call this new model $\tilde{M_1}$. Choose any metric D. Roughly, we consider the "post-surgery improvement" under Q to be $D(\tilde{M_1}) - D(M_1)$. This measures how much Q improves or degrades performance by removing it, sort of a post-training ablation study.

4 Experiments

Here we present qualitative and quantitative results for each of the three experiments in the following order: grokking (Sect. 4.1), β sweep (Sect. 4.2), mechanistic interpretation (Sect. 4.3).

4.1 Irreversible divergence in DPO

Here, we evaluate DPO over an extensive training horizon to see if it ever groks, i.e., if after a period of overfitting, the model suddenly learns to generalize (in this case, generalization means implementing a perfect identity function). Nanda et al. (2023) show grokking happens in SFT through a phase change from memorization to generalization. We find this does not happen in DPO.

Fig. 1 shows various metrics over the course of 100 epochs of DPO (and SFT, plotted for comparison). We see that DPO diverges towards a policy producing out-of-distribution actions 80% of the time, despite π_{ref} outputting nearly no OOD responses. A similar pattern holds for sequence error. Interestingly, DPO exhibit near-monotonic improvement in margin and classification accuracies, both quantities that depend on DPO's ability to act as an implicit reward function (i.e., higher margin/accuracy = more probability towards preferred sequences compared to dispreferred ones). Clearly, DPO learns that $\pi(y_w \mid x) > \pi(y_l \mid x)$. Given this, we hypothesize that DPO learns $\pi(y_{\text{OOD}} \mid x) > \pi(y_w \mid x) > \pi(y_l \mid x)$; i.e., a correct ordering of the preference pairs, but one that is

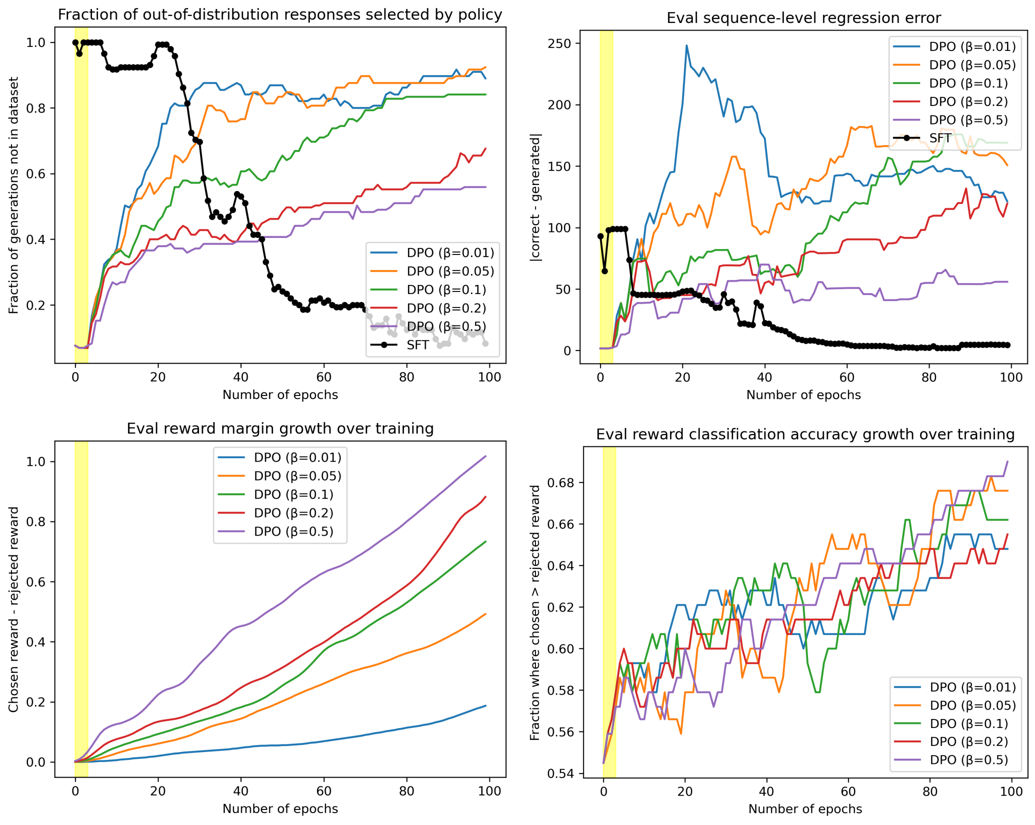

Figure 1: DPO does not grok. **Top row:** SFT and DPO training runs overlaid on the same plot. Yellow region on all plots indicates part of training where DPO stays in-distribution. **Bottom row:** DPO reward evaluation metrics improve over training, despite poor generalization.

still dominated by OOD actions. This holds even after the first few DPO epochs; we do not need all 100 to see this effect (the long training run was useful for noting the lack of grokking).

Therefore, we argue that unlike in the SFT case, grokking does not happen DPO. During both types of training, progress is made even when metrics stagnate (that is, internally, the model is likely moving towards a simpler solution due to regularizing effects from the optimizer, as hypothesized by Nanda et al. (2023)). But in the SFT case, this progress eventually leads us to a generalized solution; in the DPO case, this progress takes us increasingly out-of-distribution. While this is a sythethic experiment, we argue that this reasoning holds in other uses cases as well. Other research (Rafailov et al. (2024), Park et al. (2024)) has shown similar OOD effects despite reward metrics improving - the same pattern we observe here. We hypothesize that in those cases as well, grokking is unlikely, since the model's gradient updates continually push it OOD instead of towards a generalizing solution.

4.2 In-distribution policies have similar KL divergences

Though DPO does not grok, it is still a very useful algorithm for alignment. This raises the question of "when does DPO work?" i.e., under what conditions are DPO policies "good" (at least, not OOD). In this section, we argue that optimal policies lie in the same general KL region, regardless of how we get there. It is known that DPO produces policies whose KL is strongly correlated with $\log \beta$ and the number of update steps (Rafailov et al., 2024). Let N be the number of DPO update steps so far. In theory, given two policies A, B with different β parameters, with large enough N we can find some checkpoints A', B' so that $\mathrm{KL}(A')$ is close to $\mathrm{KL}(B')$. Because A' and B' likely come at different β, N values, there is no reason a-priori to believe that the behavior of A' and B' would be the same.

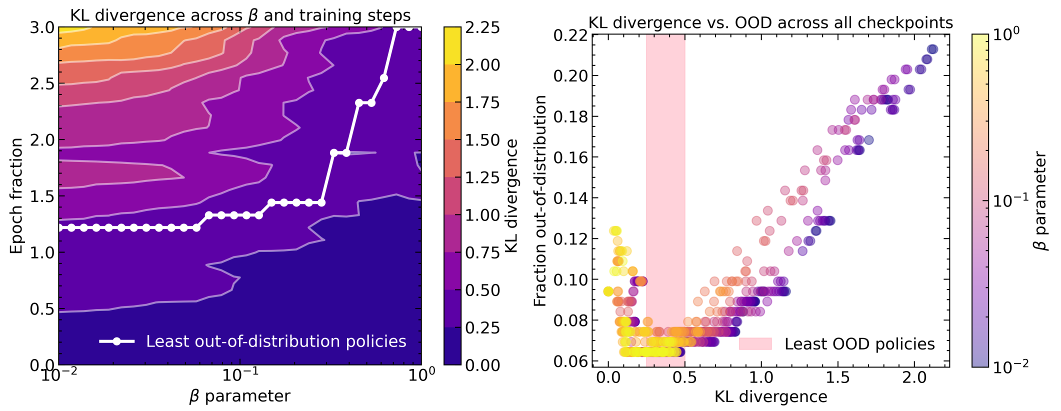

Figure 2: OOD-optimal policies have similar KLs. **Left:** KL divergence evaluated along the 30×30 (β, N) sweep. White line indicates KL of policies that minimize OOD for each β. **Right:** OOD effects as a function of KL divergence. Pink region corresponds to the KL level set from the left.

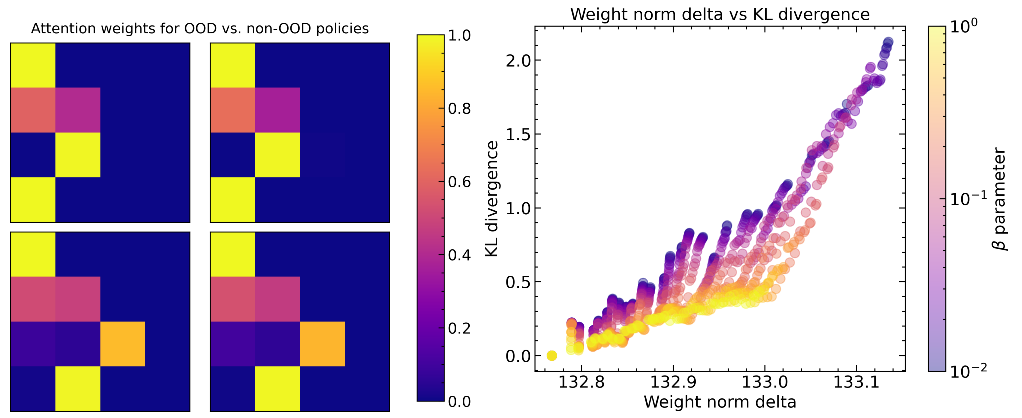

Figure 3: Weight deltas do not fully explain OOD behavior. **Left:** attention maps for OOD (left, policy A) and non-OOD (right, policy B), computed on a prompt for which A fails to give an in-distribution response, but B produces a correct one. **Right:** WeightDelta metric vs KL divergence.

While we do not claim anything about A', B' directly, we argue that the *optimal*[2] policies produced by training A and B, will, in general, have similar KLs. That is, while training A, we will produce some A^* along the way that minimizes OOD effects. Similarly, there is always some B^* that is the least out-of-distribution out of all B's checkpoints. We give empirical evidence that A^* and B^* have similar KLs. Within our fairly large sweep, all optimal checkpoints M^* have similar KLs.

To see this, we plot the KL divergence across our 30×30 (β, N) sweep in the left side of Fig. 2. We plot the optimal policies M^* for all combinations of (β, N), noting that this line stays in a bounded KL region (from 0.25 to 0.5). The right side of Fig. 2 further confirms this. We see that KL divergence and the OOD metric have a U-shaped relationship, with the best policies across all β values falling in the bottom of this curve. Practically, this means that for most reasonable β, N choices, we have some empirical reason to believe that KL will fall in a consistent range.

4.3 OOD behavior stems from attention head divergence

Now that we have established the relationship between KL and OOD policies, it is natural to ask what causes OOD behavior and KL to be so tightly linked. Here, we mechanistically explore what DPO does to model weights, comparing an OOD policy (policy "A", with $\beta = 0.01$) with an in-distribution policy (policy "B", with $\beta = 1.0$). Consider KL a resource to be spent by DPO, since the KL term in

[2]Optimal in the sense that the OOD metric defined earlier is minimized.

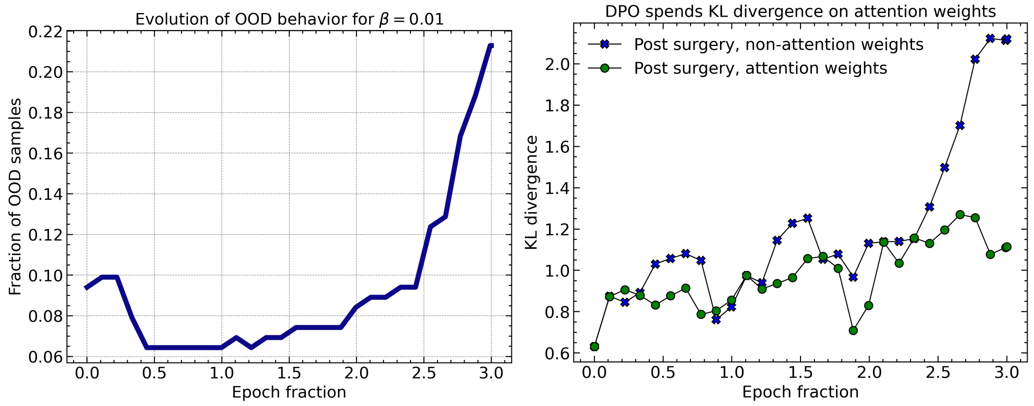

Figure 4: KL divergence is transferred to attention weights. **Left:** training step vs. OOD metric for $\beta = 0.01$ (policy A). **Right:** KL divergence after model surgery. Green is KL if we substitute out the attention weights, blue is KL with non-attention weights substituted out.

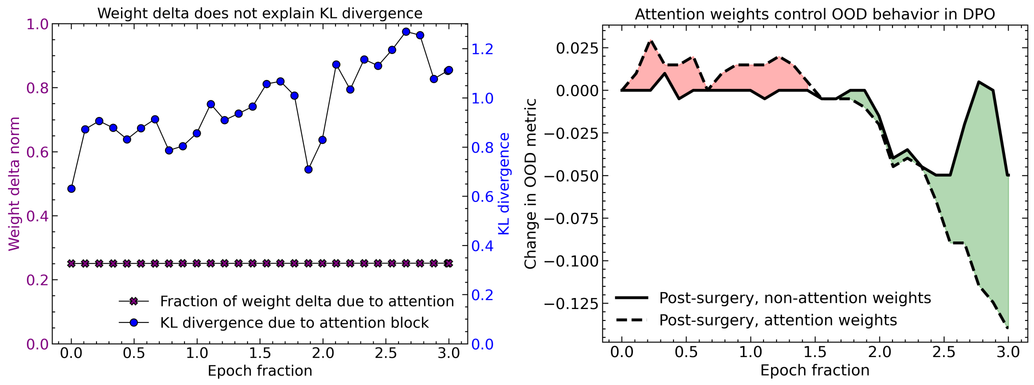

Figure 5: Attention weights are targeted by DPO. **Left:** purple line is fraction of weight change from π_{ref} in the attention block, blue line is the post-surgery KL in Fig. 4. **Right:** Dotted/dashed lines are the same policies as in Fig. 4, but we plot $\text{OOD}(A') - \text{OOD}(A)$ instead of $\text{KL}(A')$. Red region indicates where attention weight surgery harms performance, green indicates where it helps.

Eq. 1 means we only have so much ability to deviate from π_{ref} (Azar et al., 2023). We argue that 1) KL is more helpful in understanding DPO's modifications than standard weight deltas, 2) DPO spends its KL budget on changing attention head weights, and 3) OOD effects arise when attention weights deviate significantly from π_{ref}. Note that attention weights are all parameters associated with the multi-headed attention module, i.e., linear projections included[3].

One natural way to understand DPO's modifications is to compute norm-based weight deltas (Eq. 7). However, we argue that this is insufficient. In the left half of Fig. 3, we see that both A and B compute qualitatively similar attention maps, even when their resulting samples are vastly different. This is quantitatively reflected in the right half of Fig. 3, where the same weight delta induces different KL differences depending on β (and as established in previous sections, KL is important when thinking about OOD effects - OOD policies generally have high KL and vice versa).

Instead, by performing an ablation-like study via model surgery, we see that DPO spends its KL budget on attention weights. That is, attention weights contribute most to the KL difference between OOD and non-OOD policies. To see this, we substitute the attention weights in A with those from B. That is, we replace the attention weights in the OOD model with the attention weights from the non-OOD model. Let this new model be A'. We then compute $\text{KL}(A')$. In the right half of Fig. 4, this is the green curve; it grows slowly and does not match A's divergent OOD behavior (left half of Fig. 4). *Replacing the OOD attention weights with non-OOD attention weights tames the KL*

[3]Attention weights contribute around 65% of the total parameter count (65K/104K).

explosion associated with OOD behavior. We say that the green curve is the KL divergence *attributed* to the OOD attention weights, since we control for the effect of all other parameters via surgery.

Substituting out the non-attention weights instead gives us the blue curve in the right half of Fig. 4. This curve explodes- that is, replacing the non-attention OOD weights with weights from the non-OOD policy doesn't do much to combat the OOD behavior. This result is reflected in the right half of Fig. 5, in which we do a similar surgery operation but consider how OOD behavior changes instead of KL. If we replace the attention weights in the OOD policy (dotted line), then we get a dramatic dip in the OOD behavior of the modified model. But if we replace the non-attention weights in the OOD policy (solid line), we do not see much change in model behavior.

One question remains- is this sensitivity to attention weights simply due to larger gradient updates to these weights? In the left half of Fig. 5, we argue no. Between each successive checkpoint, the fraction of weight change in the attention block is only ever 25% of the total weight change from π_{ref}, indicating that DPO does not apply disproportionately larger updates to the attention block. Even so, the KL divergence attributed to the attention block via the surgery experiments indicate that these weights are most heavily modified by DPO. This leads to an interesting conclusion: DPO somehow selectively updates the attention weights without applying large-magnitude updates. It is an overabundant amount of these attention weight updates that causes OOD behavior.[4]

5 Conclusion

Via a toy dataset and several sets of experiments, we study the OOD problem in DPO at a macro and mechanistic level. We show that due to OOD effects, DPO does not exhibit grokking. To explain this, we hypothesize that both preferred and dispreferred sequences are assigned lower probability than out-of-distribution trajectories. Next, we show that OOD effects are minimized in a particular KL region across different β values. Finally, we analyze what DPO actually does to models to push them OOD, arguing that DPO imparts high KL to attention blocks specifically, causing policies to go OOD.

The primary limitation of this work is that it only considers a specific toy setting. It would be very interesting to see if these theories hold up in other settings, particularly real-world ones with billion-parameter models and larger datasets. However, we hope we have provided an insightful analysis into an open research question with large real-world impact. Additionally, the analysis tools presented here (KL/OOD model surgery in particular) may be useful for other interpretability studies.

By doing this project, I got really comfortable with how Transformers and DPO works. Through fixing many bugs in my implementation, I also learned a lot about how to debug and prototype models. It was very fun to think deeply about why DPO behaves the way it does, and try to verify my hypotheses in a quantitative and scientific way.

6 Ethics Statement

DPO is likely the last algorithm an LLM "sees" before it hits real-world users. As a result, any work surrounding DPO has the ethical risk that comes with working with algorithms interfacing with users. Specifically, by applying the methods in this paper and exploiting DPO's OOD behavior, a malicious actor could figure out how to fine-tune/jailbreak safety guidelines, or embed hidden attacks by forcing OOD behavior that bypasses security measures. Mitigating this risk requires further technical progress in interpretability, since understanding LLMs help us to train ones that can't be fooled by downstream fine-tuning. This could also be mitigated by strict red-teaming and testing protocols for trained LLMs deployed to users, designed to flesh out any mistakes/malicious content.

Another possible ethical issue is that DPO is very data-hungry, requiring high-quality preference data and human annotations. This data isn't always ethically sourced, sometimes coming from shops where workers are underpaid and exploited. This research specifically focuses on DPO OOD effects, which could potentially be mitigated by including more quality data, further incentivizing bad actors to exploit workers. To mitigate this, there needs to be strict, high-level policy regarding data collection (hopefully at the government level) that counteracts the growing data-hunger with ethical guidelines for human data collection.

[4]This conclusion would have been difficult to draw from the weight delta argument alone.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. 2024. Direct language model alignment from online ai feedback.

Shawn Im and Yixuan Li. 2024. Understanding the learning dynamics of alignment with human feedback.

Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. 2022. Towards understanding grokking: An effective theory of representation learning.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization.

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, Bradley Knox, Chelsea Finn, and Scott Niekum. 2024. Scaling laws for reward model overoptimization in direct alignment algorithms.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model.

Fahim Tajwar, Anika Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *ArXiv*, abs/2404.14367.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.